UNITED STATES UTILITY PATENT APPLICATION

FOR

A METHOD AND APPARATUS FOR RESUMING OPERATIONS FROM A LOW LATENCY WAKE-UP LOW POWER STATE

Inventors:

Opher Kahn
Doron Orenstein

42390.P8517

BLAKELY, SOKOLOFF, TAYLOR & ZAFMAN

12400 Wilshire Boulevard, Seventh Floor
Los Angeles, California 90025-1026
(408) 720-8598

EXPRESS MAIL CERTIFICATE OF MAILING

"Express Mail" mailing label number EL143554720US
Date of Deposit February 14, 2000

I hereby certify that this paper or fee is being deposited with the United States Postal Service "Express Mail Post Office to Addressee" service on the date indicated above and is addressed to the Commissioner of Patents and Trademarks, Washington, D.C. 20231.

Richard Pask
(Typed or printed name of person mailing paper or fee)

Richard Pask 2/14/00
(Signature of person mailing paper or fee)

Ins A1

A METHOD AND APPARATUS FOR RESUMING OPERATIONS FROM A LOW LATENCY WAKE-UP LOW POWER STATE

BACKGROUND

1. Field

The present disclosure pertains to the field of power management for a processing system and particularly to resuming operations upon exiting a low power state.

2. Description of Related Art

Power management is an increasingly important feature in systems such as computer systems. However, users are generally less interested in power conservation features that significantly impact the response time and performance of their systems. Thus, implementing low power states with low latency resumption is desirable.

Sub A2

The Advanced Configuration and Power Interface (ACPI) Specification 1.0b (Revision 1.0b of this open industry specification is available at http://www.teleport.com/~acpi/) provides a uniform set of definitions, power management states, and the like for implementing power conservation features in a computing system. The ACPI Specification defines the S1 power state as a low latency power state in which all system context is maintained (see § 9.1.1).

A typical system arrangement asserts the STPCLK# pin and waits for the processor to enter the stop grant state. At this point, the system may or may not shut down clocks to

the processor and/or to external buses or other circuitry. Prior art systems place system memory into a self-refresh or a suspend-refresh state. Refresh is maintained by the memory itself or through some other reference clock that is not stopped during the sleeping (S1) state.

Sub A3

One example of a memory architecture is the Rambus™ memory architecture available from Rambus Corporation of Mountain View, California. Some Rambus™ parts offer various power conservation modes (see "Direct Rambus™ Memory for Mobile PCs" also available from Rambus Corporation). Active, nap, standby, and PwrDown (powerdown) modes are available. A Rambus™ Dynamic Random Access Memory (RDRAM) automatically transitions to standby mode at the end of a transaction. When a memory transaction request is sent out to the memory array, the appropriate RDRAM device exits standby and services the request.

Sub A4

Power consumption may be further reduced by placing RDRAMs in a nap mode or a powerdown mode. Nap and powerdown modes may be entered by sending commands to the memory. From both the nap mode and the powerdown mode, a resynchronization time is required by the RDRAM for memory system's delay locked loop to synchronize the RDRAM interface to the channel clock.

Unfortunately, these memory power conservation states do not directly map into power conservation states enumerated by the ACPI Specification. The system designer is left to determine which states to use at what time, and how to enter and exit such states. Particularly puzzling is how to perform re-initialization in such a memory architecture when exiting low latency states such as the ACPI S1 state where nap or powerdown memory power conservation modes are likely to be used.

When exiting nap and powerdown states, the memory requires clock re-initialization. Exiting the ACPI S1 state, however, typically results in returning directly to operating system code (in the memory subsystem) after deassertion of the STPCLK# interrupt. Accordingly, it may not be possible to execute low level software such as BIOS software in response to a transitions out of the S1 state. Consequently, the memory subsystem may not be sufficiently re-initialized to allow memory accesses to commence.

Brief Description of the Figures

The present invention is illustrated by way of example and not limitation in the figures of the accompanying drawings.

Figure 1 illustrates one embodiment of a system utilizing presently disclosed techniques.

Figure 2 is a flow diagram for operations of one embodiment of the system of Figure 1.

Figure 3 illustrates additional details of initialization operations performed by one embodiment.

Detailed Description

The following description provides a method and apparatus for resuming operations from a low latency wake-up low power state. In the following description, numerous specific details such as memory types, signal names and logic partitioning/integration choices are set forth in order to provide a more thorough understanding of the present invention. It will be appreciated, however, by one skilled in the art that the invention may be practiced without such specific details. In other instances, control structures and gate level circuits have not been shown in detail in order not to obscure the invention. Those of ordinary skill in the art, with the included descriptions, will be able to implement appropriate logic circuits without undue experimentation.

Some embodiments advantageously allow initialization sequences to be performed in conjunction with exiting low latency low power states. This initialization may be important to ensure proper operation of memory subsystems that have complex initialization requirements. Some embodiments perform such initialization using hardware to avoid any reliance on the memory subsystem as the memory subsystem may not yet be operational. Likewise, some embodiments perform the initialization operations in a manner transparent to the operating system and/or to the central processor.

Figure 1 illustrates one embodiment of a system utilizing presently disclosed techniques. The system of Figure 1 includes a memory interface 110 and a hub interface 102 which are coupled to a processor 140 and a graphics engine 150. As indicated by a

dashed box 120, the processor 140, the memory interface 110, the hub interface 102 and the graphics engine 150 may be integrated into a single component. Also within the component is a clock and power unit 106 which may control global clocking a power supply for the memory interface 110 , the processor 140, and the graphics engine 150. The memory interface 110 is also coupled to a memory subsystem 160. The memory subsystem 160, when the system is operational, contains an operating system 170. The hub interface 102 is coupled to an I/O control hub (ICH) 180. The ICH 180 is coupled to generate a stop clock signal (STPCLK#) on a signal line 186 for the processor 140. The ICH 180 is also coupled to a secondary bus 185 which has coupled thereto a memory device 190 storing BIOS routines.

The embodiment of Figure 1 utilizes logic in both the ICH 180 and the memory interface 110 to exit from a low latency low power state and to appropriately initialize a memory subsystem 160 so that execution may be resumed. Accordingly, the memory subsystem 160, the operating system 170 and the processor 140 need not be involved.

The operation of one embodiment of the system illustrated in Figure 1 is illustrated in the flow diagram of Figure 2. In block 200, low power state entry conditions are detected. Detecting conditions such as inactivity or low battery power which warrant the entry into a low power state may be performed by hardware, software, or combination of both. Known or otherwise available techniques may be used to do so. Additionally, a user request such as a shutdown command, a special power button, a hot key, or keyboard or lid closure may trigger low power state entry. Once it is determined that a low power state should be entered, the system also typically selects which of a set of low power states to enter. For example, the ACPI specification describes low power

system states S1-S5. In this case, a low latency, low power state such as the ACPI S1 state is selected. A low latency, low power state maintains system context and allows a relatively rapid resumption of system execution. As such, complete re-initialization of components such as the memory subsystem 160 may not be feasible or desirable in a low latency, low power state.

Next, in block 205, memory control logic 135 places the memory subsystem 160 in a self-refresh mode. Since complete re-initialization may not be performed upon resumption from the low latency, low power state, the hardware maintains at least some of the configuration information determined at boot time as indicated in block 210. To maintain this information (typically stored in registers within the memory interface 110), power is maintained to the memory interface during the low latency, low power state. Additionally, as indicated in block 212, a bit may be set indicating which low power mode is being entered. The BIOS may indicate the power state by writing a value stored in the memory interface 110. In other embodiments, such a value may be set using other software or hardware techniques and may be located elsewhere, such as in the ICH 180. Additionally, the bit may be set in a different order with respect to other low power state entry operations so long as the bit is set before the low power state is actually entered.

The global clock and power unit 106 shuts down clocks in an appropriate order to prevent malfunctions. For example, global S1 state logic 108 may control gating of the clock to the CPU and/or the memory interface or other components. Glitches or other malfunctions may be avoided by gating clocks to, for example, the memory interface when a phase locked loop for the component is shut down. Additionally, prior to entry into the low power state, S1 messaging logic 184 may generate a message for the memory

interface 110 to indicate that the memory interface 110 should flush any write buffers. This may alternatively be performed in conjunction with placing he memory in a self-refresh state (block 205), and may be sequenced differently in different embodiments. The memory interface 110 may then return a message via messaging logic 125 to the ICH 180 to indicate these operations are complete and that actual entry into the low latency low power state may now occur.

As indicated in block 215, the low latency, low power state is then entered. Again, the ACPI S1 state is one example of an appropriate low latency, low power state. The system remains in the low latency, low power state until a wake-up event occurs as indicated in block 220. At this point, S1 exit detect logic 182 detects the low power state exit event which occurs as indicated in block 225. In response, S1 messaging logic 184 sends a message (e.g., S1 exit) via a bus 181 (e.g., a hub link bus) to the memory interface 110 as indicated in block 227.

S1 messaging logic 125 in the hub interface 102 receives and decodes the hub link S1 state exit message from the ICH 180 and responsively enables memory resume logic 130. The memory resume logic 130 sequences through a series of initialization commands to reinitialize the memory subsystem 160 as indicated in block 230. The memory control logic 135 may have a predefined set of instructions that it typically receives from software such as the BIOS. The memory resume logic 130 may sequence through an internally generated series of such commands in order to perform the proper initialization operations. Therefore, the memory resume logic 130 may emulate a software routine that initializes the memory control logic 135. The routine emulated may be a subset of the BIOS routine used at boot when the memory is first initialized or when

another low power state is exited (e.g., the ACPI S3 state).

Since the memory resume logic 130 itself sequences through the series of initialization commands, interaction is not required from the processor 140 or the memory subsystem 160. Advantageously, this transparency allows complex initializations to be performed at a time when the memory subsystem is unavailable. Moreover, since states such as ACPI S1 typically resume directly to the operating system 170 (which is stored in the memory subsystem 160), this transparency may be required as there may be no other opportunity for the processor 140 to execute a code sequence prior to needing to access the memory subsystem 160.

Sub A5

For example, in a system which utilizes a Rambus™ memory subsystem, putting the memory subsystem in a nap state or a powerdown state requires that certain initialization operations be performed prior to the memory subsystem resuming normal operations and returning any data from memory. Thus, if a system enters the ACPI S1 state and expects to exit by executing operating system code, the memory subsystem 160 will be unavailable to retrieve that operating system code. Thus, the intervention of hardware memory resume logic 130 may allow the ACPI S1 state to be used in a system with a Rambus™ memory subsystem.

Sub A6

Details of the initialization operations performed in block 230 for one embodiment are shown in the flow diagram of Figure 3. In block 300, the memory interface control logic is initialized by the memory resume logic 130. In the embodiment of Figure 1, the memory interface control logic is the memory control logic 135. In embodiments utilizing a Rambus™ memory subsystem, the memory control logic 135 may be a Rambus ASIC Cell (RAC) which may be initialized according to Rambus

specifications (e.g., execute a RAC initialization operation and set control register values as needed). These operations are performed well after the appropriate clocks are running and stable. In block 310, the memory resume logic 130 sends a clock synchronization command and the system waits for memory subsystem clocks to synchronize. For example, the system may wait for the Direct Rambus Clock Generator (DRCG) to lock to the Clock To Memory (CTM) clock.

As indicated in block 320, the memory resume logic 130 sends a command to set a current control register to a predetermined value. For example, the current control register may be set to a midpoint value. A midpoint value may be appropriate since a low power state is being exited and complete re-calibration may be needed..The midpoint value advantageously limits the maximum number of adjustments that may need be made (maximum of one-half the total range in either up or down directions). As indicated in block 330, memory core initialization operations may then be performed. These may include a series of pre-charge and refresh operations as needed to restore the memory subsystem to normal operation.

Returning to Figure 2, the memory interface 110 sends a message to the ICH 180 after the sequence of initialization operations has been completed as indicated in block 232. This message indicates that the memory interface has completed initialization operations and is prepared to resume normal operations. Accordingly, the ICH 180 deasserts the stop clock signal on the signal line 186 to the processor 140 as indicated in block 234. Then, as indicated in block 240, normal operation resumes, and the operating system or other software can resume execution of code stored in the memory subsystem 160.

Thus, a method and apparatus for resuming operations from a low latency wake-up low power state is disclosed. While certain exemplary embodiments have been described and shown in the accompanying drawings, it is to be understood that such embodiments are merely illustrative of and not restrictive on the broad invention, and that this invention not be limited to the specific constructions and arrangements shown and described, since various other modifications may occur to those ordinarily skilled in the art upon studying this disclosure.